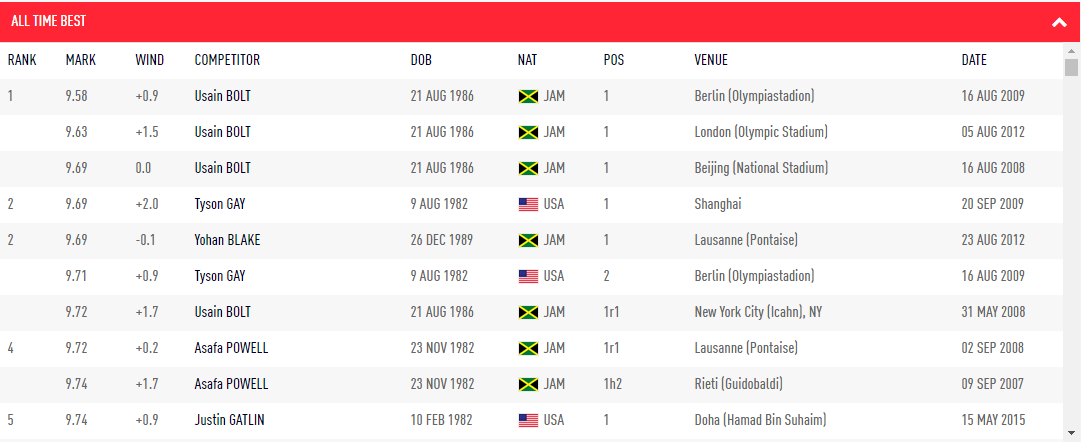
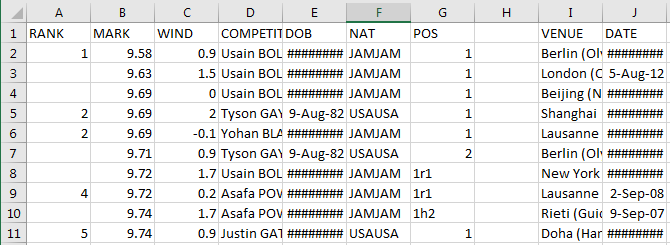
Max Wagner

Data 608 Final – Status Report

The data was pulled from the source at,[www.iaaf.org](http://www.iaaf.org), which had the initial appearance of a web table.



It was then pulled directly into a csv file, which had the initial appearance:

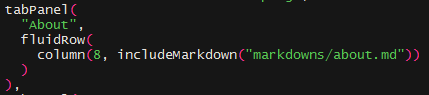


The data was cleaned using the following two R files: [1](https://github.com/maxwagner/608/blob/master/final/cleaning/cleaning.R) and [2](https://github.com/maxwagner/608/blob/master/final/cleaning/getrecords.R).

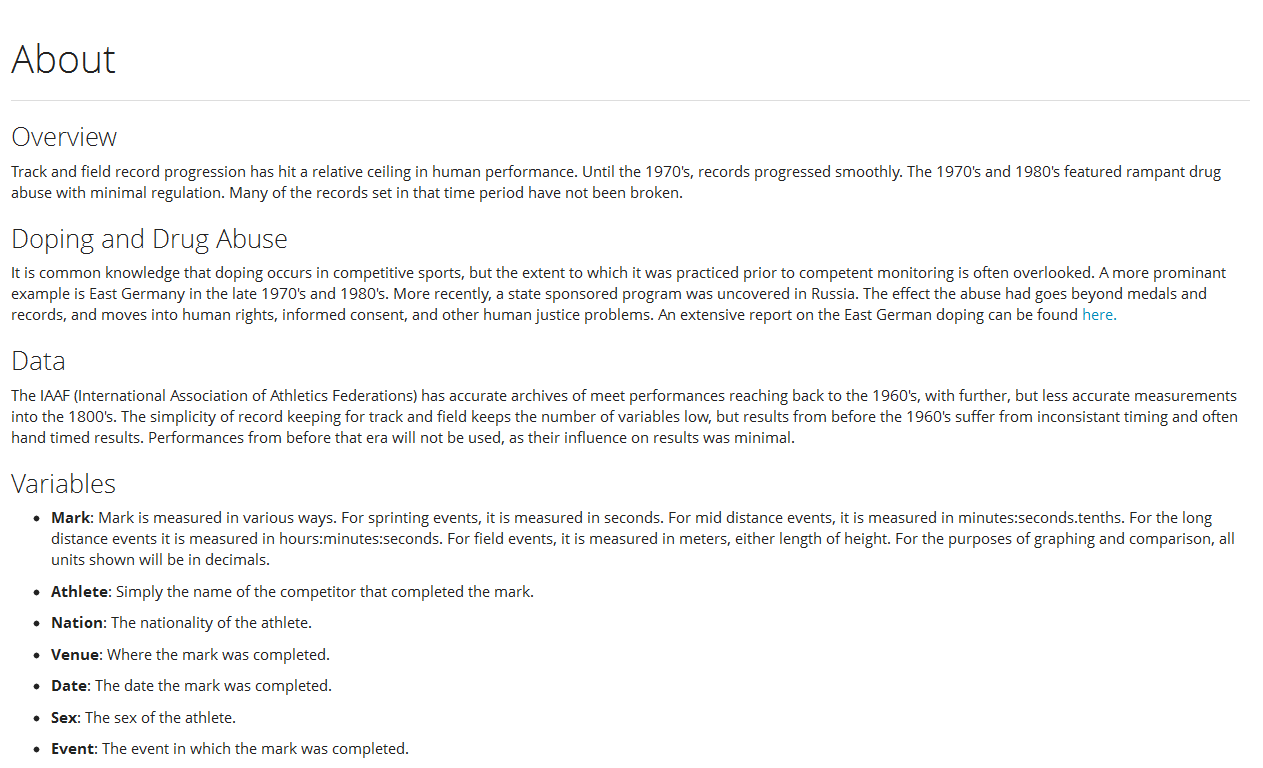
The first cleaning file removes excess columns, excess times, fixes whitespace issues, trims extra letters and numbers from entries, renames athletes, and finally combines all event csv files into one large csv. The Stringi package was helpful here to fix my capitalizing of names issue.

The second cleaning file splits the data set by type of event. This was done because the scoring is done differently depending on the event type. Each event type was then transformed accordingly. For example, the mid distance events are recorded in the format (mm:ss.dd). This was converted to a decimal format. For example, the time (02:30.00) would be converted to 2.5 minutes. This was done for all event types.

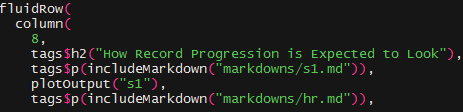
The next step was to create a shell of a shiny app to house everything I wanted to display. I knew I had separate pages I wanted to use, so I tried out tab panels in shiny. I outsourced the actual text to different markdown files to keep the main ui.R file cleaner looking.

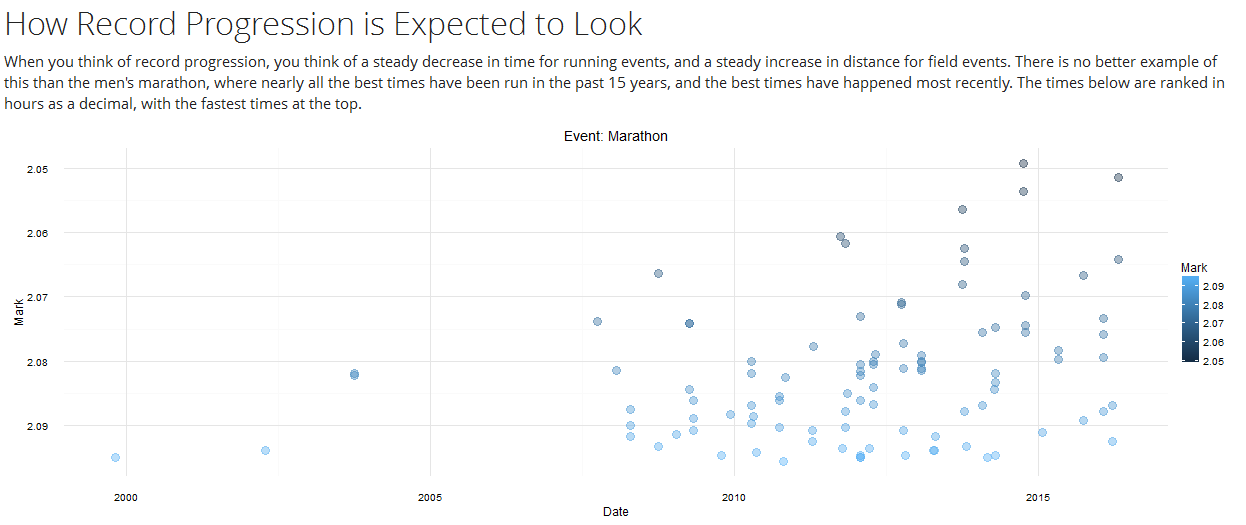


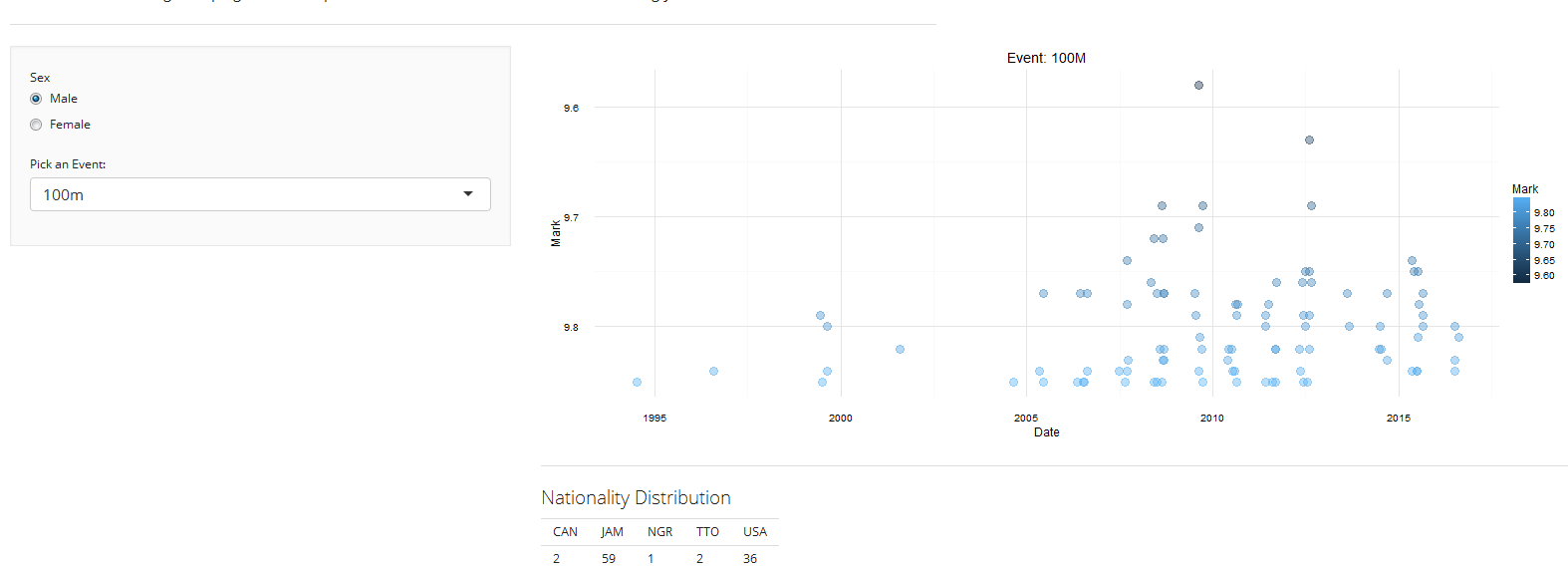
The first page I wrote up was the 2-3 paragraph summary of what was going on in the shiny app and some descriptions of the variables, and what I did to get them to that point.



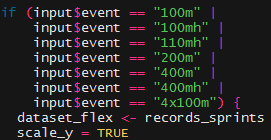
I then made a few static graphs to get a feel for the look I was going for on the next page. I ended up using a style sheet and some other options to get ggplot2 to blend into the page and have a “sleeker” look than it does normally. Fluid rows, columns, and the tags function were especially helpful to get everything to fit nicely, and adjust when the page resized.





The final part of the project was certainly the hardest. Loading every possible combination of records into a graph and table view. I considered a single dropdown, but it looked cluttered with the male and female choices. I ended up making sex a radial button choice, and the event a dropdown. It cut the clutter in events down by half. 

The next issue in the graphs I faced was flipping the y-axis for some events, and not for others. I settled on using a Boolean that was chosen in each if statement. Scale\_y, and then scale\_y\_reverse in the graph itself.

E:\Libraries\Pictures\puush\ss (2016-12-13 at 04.00.05).png

The small table underneath the graph was a surprisingly annoying bit to produce. The table() function outputs horizontally, but when put into shiny, it was initially viewed vertically, which was a mess to look at. The fix was to wrap the table function in a data frame, and then transpose it. It then printed horizontally, but unfortunately it also printed out ugly variable names. The fix for this was eventually found in the shiny function renderTable, where column and row names can be set to false.

E:\Libraries\Pictures\puush\ss (2016-12-13 at 04.04.49).png

E:\Libraries\Pictures\puush\ss (2016-12-13 at 04.06.14).png